The FLOSSWALD Information System on Free and Open Source Software

Meike Reichle & Alexandre Hanft

University of Hildesheim
Institute of Computer Science
Intelligent Information Systems Lab
D-31141, Hildesheim, Germany
meike.reichle|alexandre.hanft@uni-hildesheim.de

Abstract

We propose the implementation of an intelligent information system on free and open source software. This system will consist of a case-based reasoning (CBR) system and several machine learning modules to maintain the knowledge base and train the CBR system thus enhancing its performance. Our knowledge base will include data on free and open source software provided by the Debian project, the FLOSSmole project, and other public free and open source software directories. We plan to enrich these data by learning additional information such as concepts and different similarities. With this knowledge base, we hope to be able to create an information system that will be capable of answering queries based on precise as well as vague criteria and give intelligent recommendations on software based on the preferences of the user.

1 Introduction

In the beginning of free and open source software, these programs were mainly written for their developers' own needs or those of their communities. This has produced a large and diverse range of software wich often offers numerous alternatives for the same task, such as text editors, e-mail clients or web browsers. Also because of this existing project descriptions are mostly technically phrased and focus on the project's technological features.

However, in order to chose from a range of available software especially less experienced computer users mainly ask for qualitative attributes such as usability, stability and an agreeable look. Already existing software directories also offer mainly technically oriented search possibilities. What's missing here is the link between the user's qualitative expectations and the technical attributes of a project.

We believe it is possible to learn to translate these qualitative attributes into a set of technical features. Our plan is to design and implement an information system on free and open source software, FLOSSWALd, that offers searches by technical as well as qualitative attributes. The system's knowledge base will consist of software descriptions, improved with tags and user feedback on the results.

First, we illustrate our motivation to launch FLOSSWALD. Section 2 introduces the idea behind FLOSSWALD, including a first analysis of the available data sources. Section 3 presents related work that we have examined in the course of the project's creation. This paper closes with a conclusion and an outlook in section 4.

1.1 Motivation

Since its first public emergence, free and open source software has steadily gained more popularity. What first was an elaborated hobby among computer scientists, has today found its way to a much larger community, including less experienced computer users. With this increasing amount of users and of course also developers there has also been an increasing demand for information on that matter.

The Free and Open Source Software Community however is a complex social and technical network that consists of hundreds of thousands of individual groups and projects. Since the large majority of these projects is non-commercial, they usually don't engage in advertisement or public relations but focus on technical development and community contacts.

This has created a wide supply of free and open source software in all degrees of quality, from low class to very high grade, but most of it only known to its users, other insiders or those who know where and how to search for it. But the popularity of free and open source software is rising, and a growing number of computer users who have just recently begun using some of the most popular free and open source software such as the Firefox browser or the Thunderbird e-mail client are now considering to exchange also other programs for a free and open source alternative. These computer users usually only know very few isolated projects and don't know about the actual free and open software scene.

Free and open source software is not limited to private users though. With the increasing cost of commercial suites and the ongoing need to reduce costs, free and open source business and server applications such as the Apache HTTP server or the mysql database management system have already gained a substantial market share. The Netcraft web server survey for July 2006 [Netcraft 2006] shows a 63.09% share for the Apache web server, followed by Microsoft's IIS with 29.48%.

This rising interest in free and open source software also creates a new need for information on free and open source

software projects and their nature and quality. However, knowledge about this topic is still very much restricted to insiders who are themselves active in the Free and Open Source Software Community. While simple technical questions such as *"What database does application XY use?"* are sufficiently easy to answer using e.g. web search machines, more vague questions such as *"What is the right e-mail client for me"* or *"Which GNU/Linux distribution should I put on my small company's web-server?"* are a much harder task. Such questions don't only need technical information but also meta information such as how old or established a project is, how many users or developers it has, how mature its code base is, or how reliable it is to still be around in a few years.

Existing information services such as Freshmeat or Sourceforge rely heavily on technical criteria and language and are thus of only small use to inexperienced users. As a consequence, we plan to implement an intelligent information system that meets this need by offering more intuitive search criteria and intelligent search tools based on user preferences and learned similarities between software or user groups.

1.2 A note

In this paper we use the term "free/libre and open source software" in order to include both, the Open Source and the Free Software community. The term *free* in this text is thus not understood as *"for no cost"* but in the sense of freedom, meaning

> *[...] the users' freedom to run, copy, distribute, study, change and improve the software.* [FSF 2004]

2 The FLOSSWALD Project

First we introduce the project and describe the investigated data sources for our knowledge base: Debian packages, DebTags, Debian changelogs, the Debian bug tracking system and the FLOSSmole project followed by a disscussion of their adequacy. The last part of this section presents the planned implementation as an instatiation of a more general framework for knowledge-based systems.

2.1 Concept

FLOSSWALD, the **F**ree/**L**ibre **O**pen **S**ource **S**oft**W**are and **A**pp**L**ication **D**irectory, is a project proposal that aims to use a case-based reasoning system that includes information about the individual softwares in its knowledge base. We decided to us a case-based system, because we are dealing with vague criteria and use a large collection of individual information entities. The system is further equipped with several machine learning components that are meant to improve system performance by creating additional knowledge in the form of concepts, e.g. user groups or similarities (such as *"of a similar kind"* or *"do the same task"*), from the provided data.

2.2 The Knowledge Base

The knowledge base will be developed in three stages, each integrating a new data source. Our first data source will be the Debian GNU/Linux package repository. This repository offers several sources of information on software, which will be elaborated in the following sections. Secondly, we will extend our knowledge base to also include data from other free and open source software directories, such as Freshmeat, Sourceforge or Savannah. For these data it would be possible to cooperate with the FLOSSMOLE project [Howison et al. 2006] that provides raw data, mined at Freshmeat, Sourceforge, ObjectWeb and Rubyforge and also from donated data by other research teams. These data can be used to enrich the information already gathered in stage one. In the last stage, we hope that the FLOSSWALD project will have gathered enough momentum to also attract software authors, maintainers and users themselves and offer them a way to complete and update our data on their projects or enter new projects as they arise.

The Debian Project

Debian GNU/Linux is a free operating system, that is developed by more than a thousand volunteer developers and many more contributors such as package maintainers, translators and documentation writers all over the world, who collaborate mainly via the Internet. Debian is distinguished from other GNU/Linux distributions primarily by its overriding commitment to the principles of Free Software as laid down in its "Free Software Guidelines" [Debian 1997], its non-profit nature, and its open development model. Debian GNU/Linux is a binary Linux distribution, which means that it takes existing free software, "packages" it and provides those packages to be installed on the user's system. So instead of installing or compiling a piece of software a user downloads and installs the according package.

Debian GNU/Linux has, due to its age and popularity, probably the largest selection of prepackaged free and open source software of all GNU/Linux distributions. The Debian package repository [1] currently (as of July 2006) holds 15,660 binary packages and 9,053 source packages in its stable release. This number is still growing, for the upcoming release the package repository currently holds 17,583 binary packages and 10,228 source packages. And, what's most important, all of these packages come with a textual description, such as the example in Fig. 1.

These descriptions already offer a great wealth of information. The textual description can be analyzed with different information retrieval tools, extracting important terms or finding similar descriptions in other packages. The package's size, dependencies, section and priority can also provide conclusions on its suitability for an existing system, its nature or purpose. Additionally to this, the Debian package repository offers several other sources of information.

DebTags [Zini 2005] is a project started by Enrico Zini. Those tags are shown alongside package descriptions where available (Fig 1, last two lines) and give meta information on the software. The tags include numerous different ontologies representing different "perspectives" such as what the software is used for, what interfaces are used, what role the software has (server, client), its programming language, used protocols and many others. DebTags are in a machine readable format and thus allow for smarter search and navigation interfaces than the original full text Debian package search.

Debian further collects detailed anonymised usage data on its packages. Debian GNU/Linux users can voluntarily

[1] http://packages.debian.org

```
Package: 3dchess
Priority: optional
Section: games
Installed-Size: 136
Maintainer: Debian QA Group <packages@qa.debian.org>
Architecture: i386
Version: 0.8.1-12
Depends: libc6 (>= 2.3.6-6), libx11-6, libxext6, libxmu6,
libxpm4, libxt6, xaw3dg (>= 1.5+E-1)
Filename: pool/main/3/3dchess/3dchess_0.8.1-12_i386.deb
Size: 33564
MD5sum: fecee217870b621286f75e528496d3b1
SHA1: 88343e19f566cf5cd11ef099bad97fbabf4e316d
SHA256: 3601709708044f7e489a0a74dbe4aca0e04b2fe1bc
533655b268af36fb6abd2c
Description: 3D chess for X11
3 dimensional Chess game for X11R6. There are three boards,
stacked vertically; 96 pieces of which most are the
traditional chess pieces with just a couple of additions;
26 possible directions in which to move. The AI isn't
wonderful, but provides a challenging enough game to all but
the most highly skilled players
Tag: game::board:chess, interface::3d, use::gameplaying,
x11::application
```

Figure 1: The package description of a Debian package

install a program called *popularity contest*[2] (popcon) that sens anonymised reports to the Debian project indicating what packages are installed on a user's system and when they have been used the last time. These data allow a first take on e.g. the popularity of a particular software or – if analysed on a per user basis – what softwares are often used together.

Debian Changelogs and the bug tracking system [3]: Every Debian package has a changelog where all changes or updates on the package are noted, the Debian bug tracking system is used by developers, maintainers and users to report problems or bugs of a piece of software and monitor these reports and their solutions e.g. by patches or new versions. Both these sources can give information on the up-to-dateness and stability of a package.

The FLOSSmole Project
The FLOSSmole (formerly OSSmole) project is a collaborative project

> *[...] designed to gather, share and store comparable data on and analyses of free and open source software development for academic research.* [Howison et al. 2006, S.1]

Its aim is to create a trusted dataset for research communities dealing with free and open source software, such as the TREC dataset [TREC 2005] in the information retrieval community or the UCI repository in machine learning [Newman et al. 1998]. As a collaborative project FLOSSmole expects its users to give back any improvements or additional scripts that are created when using the provided data.

In order to achieve this the FLOSSmole project identified several key requirements [Howison et al. 2006], that it aims to comply with: Firstly, the collected data is required to be easily available, without a lengthy requisition procedure or having to deal with complicated repositories, such as versioning systems. This is achieved by offering simple unmonitored web downloads. Furthermore, the provided data has to be comprehensive and compatible, offering different timestamped versions in order to allow historic comparisons and also comparisons between different free and open source projects or repositories by including mappings between the different databases.

FLOSSmole gathers its data both by web spidering and also by using the project directory's database dumps where available. These datasets are then "cleaned": Where databases are available they have to be restructured, since they all use their own structure and attributes, the individual attributes have to be mapped to their respective counterparts in other repository's databases. Spidered data have to be checked and cleaned of false input, redundancies and the like. In the above mentioned repositories (Freshmeat, Sourceforge, ObjectWeb and Rubyforge) FLOSSmole mines different data, including project data (name, programming language, platform, operating systems, intended audience, project topics) and developer-related data (number of developers, their contact information and roles). Additionally FLOSSmole mines on the bug tracking systems of Sourceforge, Savannah, Freshmeat and the Apache Foundation, collecting information on bugs, such as when they were opened and closed, their priority and status over time.

All of these data are available as raw database dumps and may be used freely by scientific projects dealing with free and open source software.

Using these data we intend to map the respective projects to their developed software and thus extend the already existing cases with new attributes or create new we cases where necessary.

2.3 Adequacy of the Data Sources

We believe that the presented data sources are adequate for an information system such as the one we are planning. In order to serve as a knowledge base for a CBR system, the data provided needs to be correct with respect to content, of a sufficient supply as well as structured and in a simple format. Regarding the Debian packages, the correctness of the data may be assumed since they are taken directly from a working system and the Debian policy tells package maintainers to describe their packages in a neutral and objective way. Debian's open development model further supports their correctness. Also, all information on the Debian packages is freely available in a structured pure text file and can thus easily be worked with or stored, e.g. into a database. The same goes for the data provided by the FLOSSmole project which can be downloaded as a pure text database dump and are also collaboratively reviewed and updated.

Of course the knowledge base of our information system will be incomplete in the beginning. Based only on the Debian packages during the first stage it will miss software that e.g. only runs on Microsoft Windows or software that is not packaged for Debian for other reasons, e.g. because it has a license considered non-free by the Debian Free Software Guidelines. Also we will have to evaluate which attributes are provided by the respective sources, how they

[2] http://popcon.debian.org/
[3] http://bugs.debian.org/

can be mapped or converted, which of them are useful to us and whether we might have to create or extract additional information such as user ratings or associated user groups/categories.

2.4 Integration into a Knowledge-Based System

Due to the heterogeneity and the large amount of free and open source software, it seems to be appropriate to use case-based reasoning to give the user advice according to their vague descriptions and (soft) constraints. In such a case-based reasoning system, each software project should be represented by an individual case. Additionally, users should have the option to tag already known software with freely chosen tags to enrich the software descriptions with attributes that conform better with most users' level of abstraction than the mostly technical information we get from the afore mentioned sources. To finally create the "link" between more formal project descriptions and the vague descriptions from the users' perspectives we plan to use machine learning technology.

In order to query the system a flexible input mask is offered that allows giving information on what (type of) operating system is used or what other software is already installed. In order to not overload the user interface the according menus would be optional so that if the user choses to give a particular information e.g. *Browser* this would then open a drop down menu including a list of browsers to chose from. Further on, the user has the possibility do define precise as well as vague requirements, with the vague ones including points such as *"Runs on slow computers"*, *"Language can be switched"* or *"Easy to install"*. These more intuitive requirements would then be mapped to actual technical queries, e.g. *"Easy to install"* would be true if the project provides prebuilt binaries and an installing mechanism for the user's operating system. It should also be possible to prioritise those requirements. Additionally to these options the user can also give keywords for a free text search that will either be conducted over the full software information or only on selected fields. The user will be free to choose from these and probably also other options and combine them to create a query that best represents his or her information need.

FLOSSWALD is obviously a knowledge-based system, because most of the functionality FLOSSWALD should provide depends on knowledge and its processing. [Althoff et al. 2006] presented a framework for knowledge-based systems (KBS) that appears to be promising for realising FLOSSWALD. Their underlying idea is to once implement a highly flexible knowledge based system, that is then reconfigured for different use cases, instead of every time building a new system from scratch. To achieve this goal [Althoff et al. 2006] combine case-based reasoning, experience factory approaches, software product-lines and agent technology within one architecture for knowledge based systems.

Within this KBS architecture different components are responsible for usage and maintainance. This complies with our scenario where some users search the system for advice on a software while others update project descriptons or enter new ones. All of these tasks are associated with different roles within the system and are carried out by a software agent, possibly supervised by a human operator, depending on how easy the according task can be automated. This again corresponds to our estimation that also plans to first maintain the knowledge base and its content by hand but automate this process using machine learning technologies once the knowledge base is sufficiently large and enough correct training data has been created.

Mapping this KBS concept to our concept of the FLOSSWALD project would result in a setup as illustrated in Fig 2: FLOSSWALD, as user interface of the whole knowledge-based system, is located in the system tier and interacts with the users. It uses a CBR-Agent$_1$ for retrieval. The CBR-Agent$_1$ itself accesses the case base$_1$ inside the knowledge access tier. The maintainance which includes taking care of the case base and similarities for retrieval is done separately under the hood of a case factory represented by CFM Agent$_1$ in the maintenance and build-up tier. The case factory itself incorporates several roles like case manager or libriarian. It uses the services of a machine learning agent$_2$ for discovering the relationships between technical features and qualitative attributes. The CBR-Agent$_1$ and the machine learning agent$_2$ both process knowledge intensive tasks and are thus located in the knowledge worker tier.

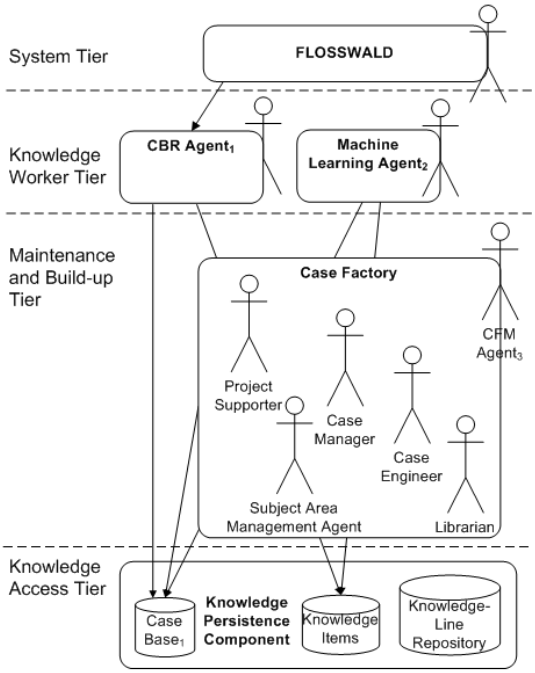

Figure 2: The FLOSSWALD concept mapped on a KBS

3 Related work

Existing information systems on free and open source software include Freshmeat, GNU Savannah,Berlios and Sourceforge. Many of these directories (e.g. Savannah, Berlios and Sourceforge) also offer development infrastructure such as web space, mailing lists, version control systems, bug tracking systems, or wikis. Those directories thus only include information on projects they also host. This has led to a certain redundancy since many projects registered with e.g. Sourceforge, so they are listed there but do not use the provided infrastructure but their own.

A system that is rather close to our concept is the FSF/UNESCO Free Software Directory which follows the same purpose. It, however, only includes software that runs on free operating systems. The FSF/UNESCO Free Software Directory is also a collaborative project, offering a web interface for users to enter and update entries. Since it did not import any data but only relies on user input the directory so far only holds around 5,000 entries. Efforts to raise this number are made, e.g. by introducing contests.

There is also a commercial information service on free and open source software, ohloh.net. It currently lists almost 4,000 open source projects. Ohloh's approach is different to that of the FSF/UNESCO Free Software Directory or ours. It mines all its data automatically and has as data source the projects' source code and version control systems. Because of this, their information is mainly focused on the projects code base. There are also a tags, that are used to display related projects. For each indexed project ohloh.net lists general information such as the age of the project, the number of developers, and its main programming language. Beside that it offers a feature called *Codebase Cost* that allows the user to calculate how much it would have cost to have that code written, based on lines of code, man years and a freely selectable yearly salary. Further on the site presents license information and several diagrams, illustrating the amount of lines of code and the activity of the projects' developers, also measured in lines of code. Because of this these numbers and diagrams only provide a measure of the effort that has been put into a particular software. Other information such as its quality or usability have to be infered from the other presented data by the user himself.

[Althoff et al. 1999] have created an information system that is designed as an experience factory and holds information on CBR technology and tools. This system is also based on a CBR system that can be queried and updated over a web interface. This system gave the original idea for the FLOSSWALD and we hope to be able to reuse some of the experiences made in the development of this system.

4 Conclusion & Outlook

We are confident that FLOSSWALD will be of great use to computer users new to free and open source software, who will most likely do vague searches, based on similarities or ratings as well as to experienced users who are searching with highly defined criteria such as required libraries or avoiding particular technologies or protocols. Our first step will be to evaluate the data provided by the Debian project and the FLOSSmole project and design a knowledge base and case structure to flexibly work with them. Then we plan to prepare and implement a CBR system based on the knowledge-based systems framework that is able to deal with the provided information and define the particular components (such as the maintenance of the knowledge base) where machine learning modules can be used to improve the system's performance.

Once the knowledge base is sufficiently large, the answer quality of the query system is satisfying and the project has hopefully gained some publicity, the last step will be to open up the project towards collaborative maintainance and implement a mechanism that allows users and software autors or maintainers to exert influence on the knowledge base itself by updating existing cases or adding new ones. To this end we will not only have to develop the technical means to actually edit the case base but also an adequate quality assurance mechanism to furtheron ensure the correctness of the data. As a basis for this we will use [Hanft and Minor 2005].

5 Sources and References

[Althoff et al. 1999] Althoff, K.-D., Nick, M., Tautz, C. (1999). CBR-PEB: An Application Implementing Reuse Concepts of the Experience Factory for the Transfer of CBR System Know-How. In Proceedings of the Seventh Workshop on Case-Based Reasoning during Expert Systems '99 (XPS-99), Wuerzburg, Germany.

[Althoff et al. 2006] Althoff, K.-D., Hanft, A. & Schaaf, M. (2006). Case Factory – Maintaining Experience to Learn. M. Göker & T. Roth-Berghofer (eds.), Proc. 8th European Conference on Case-Based Reasoning (ECCBR'06), LNCS 4106. Springer Verlag. pp 429-442.

[Debian 1997] Debian Project (1997). The Debian Free Software Guidelines (DFSG).
http://www.debian.org/social_contract#guidelines
last visited: 07/23/2006

[FSF 2004] Free Software Foundation, Inc (2004). The Free Software Definition at http://www.fsf.org/licensing/essays/free-sw.html.
last visited: 07/25/2006

[Hanft and Minor 2005] Alexandre Hanft, and Mirjam Minor. A Low-Effort, Collaborative Maintenance Model for Textual CBR. In Steffi Brninghaus (eds), ICCBR 2005 Workshop Proceedings, pages 138 – 149, August 2005, DePaul University, Chicago, USA.

[Howison et al 2006] Howison, J., Conklin, M., Crowston, K. (2006). FLOSSmole: A Collaborative Repository for FLOSS Research Data and Analyses. International Journal of Information Technology and Web Engineering. 1(3). July-September, 2006. pp 17-26.

[Netcraft 2006] July 2006 Web Server Survey at http://news.netcraft.com/archives/2006/06/28/july_2006_web_server_survey.html. last visited: 07/25/2006

[Newman et al 1998] Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases at http://www.ics.uci.edu/∼mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science. last visited: 07/25/2006

[TREC 2005] The Fourteenth Text REtrieval Conference (TREC 2005) in Gaithersburg, Maryland, National Institute of Standards and Technology (NIST), at http://trec.nist.gov/pubs/trec14/t14_proceedings.html. last visited: 07/25/2006

[Zini 2005] Zine, E. (2005). A cute introduction to Debtags at http://debtags.alioth.debian.org/paper-debtags.html. Last visited: 07/25/2006